**A Comprehensive Guide for Reading and Writing JSON with Python**

The [json](https://docs.python.org/3/library/json.html) module enables you to read JSON object from a file or HTTP response and write it to a file. It is worthwhile to spend a little bit of time to understand a few key functions that are often used for ingesting json data.

In fact, if you understand these three functions below, you are pretty much set for any JSON data ingestion.

* json.load()
* json.loads()
* json.dumps()

Before getting into actual code, let’s see what each function does.

**json.load()**

The load() function read a json file and returns a json object. In Python, it is a dictionary. Technically speaking, it deserialises a json object into a Python object, dictionary, by using the conversion table.

You first need to read the json file. If you convert it to a string and write it to a file, the file contains the string version of dictionary.

[cc lang="python" tab\_size="4" lines="-1"]

import json

f =open('/tmp/json\_pretty\_test.json', 'r')

json\_obj = json.load(f)

# or

with open('/tmp/json\_pretty\_test.json', 'r') as handle:

json\_obj = json.load(handle)

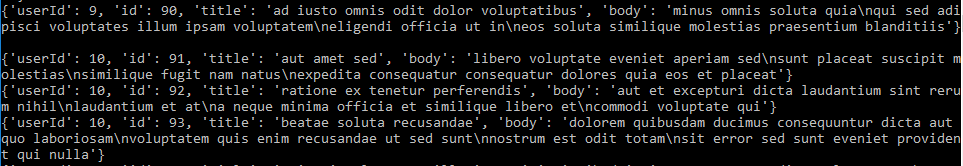
print(type(json\_obj))

for line in json\_obj:

print(line)

[/cc]

As you can see in the output, it is dictionary, not JSON.



**json.loads()**

‘s’ means string. It takes string as an input instead of file object. Type comes out as dictionary. Just like the load() function, it deserialises a json string into a Python object, dictionary.

[cc lang="python" tab\_size="4" lines="-1"]

string = '{"id":1123,"name":"John"}'

json\_obj = json.loads(string)

print(json\_obj)

print(type(json\_obj))

[/cc]

**json.dumps()**

The dumps() function returns a string representing a json object from Python dictionary object. Once you load a json file with json.load(), you can pass the resulting object to json.dumps() to get the string representation of json, which is can be written to a json file.

This is the opposite of load or loads. It serialise the dictionary into json string by using the conversion table. The conversion table maps Python object to json object. For example, None becomes null, True becomes true, single quote becomes double quote and so on.

Note that json.dumps() does not work on entire json\_obj. You need to apply it to each dictionary in the for loop.

[cc lang="python" tab\_size="4" lines="-1"]

with open('/tmp/json\_pretty\_test.json', 'r') as handle:

json\_obj = json.load(handle)

for line in json\_obj:

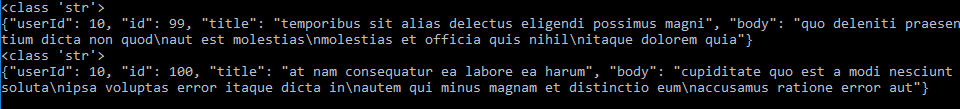
dumped = json.dumps(line)

print(type(dumped))

print(dumped)

[/cc]

As you can see in the output, the data type is string. Python data structure for dictionary is converted to json data structure (double quote instead of single quote etc).



You can pretty print Json by adding the indent argument to json.dumps().

[cc lang="python" tab\_size="4" lines="-1"]

json.dumps(line, indent=4)

[/cc]

**Code Examples**

Now that we got the basics, let’s have a look at the usage of these functions in the context of REST API data ingestion. We are using <https://jsonplaceholder.typicode.com/> which provides free API endpoints for example JSON data. The site is great for testing or experimenting with JSON over REST API. To make API calls, we are using the [requests](http://docs.python-requests.org/en/master/) module.

1. **Using json.loads() and json.dumps()**

The requests module has json() function which deserialise the JSON object in the response into a dictionary. The json\_ph\_api() returns JSON data in the dictionary format.

write\_json() uses json.dumps() to serialise each dictionary object returned from json\_ph\_api() into a json string and write it to a file.

Write\_pretty\_json() creates a pretty json file by using the indent argument in json.dumps().

json\_file\_check() prints the first 3 lines of the output from json\_ph\_api().

[cc lang="python" tab\_size="4" lines="-1"]

[/cc]

1. **Reading and writing JSON file with json.load()**

We can read the json file created above and write it to another json file with write\_json(). The file object needs to be passed into json.load() before writing.

[cc lang="python" tab\_size="4" lines="-1"]

with open('/tmp/json\_pretty\_test.json', 'r') as handle:

j = json.load(handle)

json\_file\_check(j)

write\_json(j, '/tmp/json\_test2.json')

[/cc]

1. Example of json.load()

The response is converted to a String object with .text in this example. The string object needs to be converted into a dictionary with json.load() and printed in the console with json\_file\_check().

[cc lang="python" tab\_size="4" lines="-1"]

[/cc]